IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Dhruva Ranjan Chakrabarti	§	Art Unit:	2192
		§		
Serial No.:	10/698,509	§	Confirmation No.:	9606
		§		
Filed:	10/31/2003	§	Examiner:	Thuy Chan Dao
		§		
For:	RUN-TIME PERFORMANCE WITH CALL SITE INLINE SPECIALIZATION	§	Atty. Dkt. No.:	200314557-1 (HPC.0714US)

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REPLY BRIEF

Sir:

The following sets forth Appellant's Reply to the Examiner's Answer dated September 17, 2009.

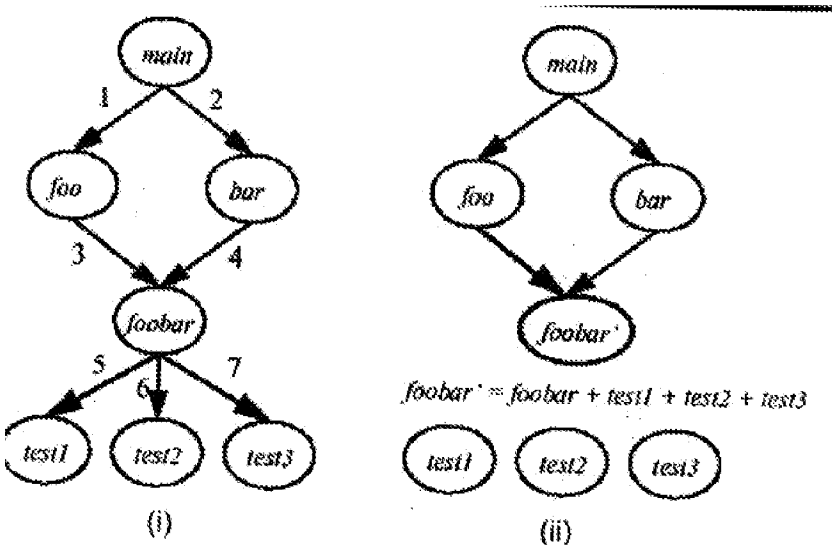
A. REPLY TO EXAMINER'S ANSWER REGARDING THE § 103 REJECTION OF CLAIMS 1, 7, 13

In the Examiner's Answer, the Examiner continues to misconstrue the term "inlining" as used in the following clause of claim 1:

> given a call-graph, if multiple call-chains in the call-graph have a common call site, **inlining** the common call site in one or more of the call-chains, without **inlining** the common call site into all of said multiple call-chains having the common call site.

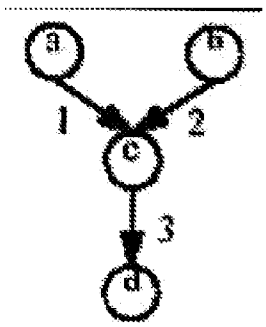
In a reproduction of Fig. 1E of Kramskoy on page 9 of the Examiner's Answer, the Examiner identified each of nodes 1072, 1080, and 1084 (which represent respective **code fragments**) as being a "common call site." This interpretation of nodes representing code fragments as being call sites is contrary to the understanding of a person of ordinary skill in the art, when "call site" is properly construed in view of the specification of the present application. For example, in the discussion accompanying Fig. 2B of the present application, the "nodes" of a call graph are referred to as corresponding to program routines, while the edges of the call graph correspond to call sites. Specification, p. 7, lines 22-23.

Fig. 2B of the present application is reproduced below:



In the Fig. 2 example, the "call sites" are labeled 1, 2, 3, 4, 5, 6, and 7. *Id.*, p. 7, lines 23-29.

Similarly, in another figure (Fig. 6(b) of the present application), call sites are identified as being the edges between nodes (*id.*, p. 26, line 25-p. 27, line 6):



(b)

In the example of Fig. 6(b) reproduced above, call site 3 would constitute one example of the "common call site" recited in claim 1.

The Examiner further argued that Kramskoy discloses "inlining the common call site 1072 in the call-chain 1068 → 1070 → 1072 → 1080 but not in the other call-chain 1068 → 1070 → 1072 → 1078; or inlining the common call site 1080 in the call-chain 1072 → 1080 → 1083, but not in the other call-chain 1072 → 1080 → 1082." Examiner's Answer at 10.

Based on the above allegation, it appears that the Examiner is arguing that "inlining" means that a code fragment (such as code fragment 1072 in Kramskoy) is in a particular call-chain. This interpretation of "inlining" contradicts what a person of ordinary skill in the art would understand the term to mean, in the context of the present invention.

As understood by persons of ordinary skill in the art and explained in the Background section of the present application, inlining "replaces a call site with the called routine's code." Specification, page 1, line 34 – page 2, line 1. As further explained by the present specification,

"in-line substitution eliminates call overhead and tailors the call to the particular set of arguments passed at a given call site." *Id.*, page 2, lines 1-3. The compilation of code fragments performed in Kramskoy, and in particular, the compilation of code fragments of a dominant path performed by Kramskoy, does not constitute inlining a common call site of a call-graph in one or more call-chains of the call-graph, as recited in claim 1.

In fact, Kramskoy itself provides objective evidence that the discussion accompanying Fig. 1E of Kramskoy, in which code fragments of a dominant path are compiled, is not the same as inlining, as recited in claim 1. Appendix 7 of Kramskoy in column 85 provides an example of inlining in the context of Fig. A7-3. *See* Kramskoy, 85:46-56. Kramskoy states that the "Method bar has been **in lined so that bar is now <u>contained</u> in the section 4034 as section 4038**." *Id.*, 85:53-56 (emphasis added). As further explained in column 84 of Kramskoy, "the Method bar is moved to be in line with the rest of the code of public int foo (int) from which bar is called," to perform inlining. *Id.*, 84:13-27.

The example given in column 84 is as follows:

```
Class X
    public int foo (int)
        {
        :
        // call bar ( )
            {
            // body of bar from class X
            :
            }
        ;
        }
```

In this example, the body of the Method bar has been incorporated into the code for int foo, to perform inlining. Thus, Kramskoy has clearly made a distinction between inlining and merely compiling code fragments, as shown in Fig. 1E of Kramskoy.

The Examiner also cited the following passages of Kramskoy as purportedly supporting the rejection: column 6, lines 47-49; column 11, lines 11-13. The cited column 6 passage of Kramskoy indicates that "[w]here the fragments are compiled, preferably the compiler **is able** to optimize the complied code[, where such] optimizations might include inlining." The cited column 11 passage of Kramskoy states that "[c]ompiling dominant paths of execution **allows** loop optimizations and inlining to be performed." Both these passages merely indicate that inlining is a separate process (for performing optimization) in a compiled dominant path. These cited passages of Kramskoy do not support the Examiner's allegation that the compilation of the code fragments in the dominant path of Fig. 1E of Kramskoy constitutes inlining.

In fact, Fig. A7-3 of Kramskoy shows an example of inlining performed on **compiled code**. Thus, it is clear that any inlining performed in Kramskoy would be **after** the dominant path compiling depicted in Fig. 1E. Thus, Kramskoy itself provides objective evidence that the compilation of code fragments in a dominant path does **not** constitute inlining.

In the section of Kramskoy that actually discusses inlining in detail (Appendix 7), there is absolutely no teaching that the inlining process actually contemplated by Kramskoy involves the following:

> given a call-graph, if multiple call-chains in the call-graph have a common call site, **inlining** the **common call site** in one or more of the call-chains, **without inlining** the **common call site** into all of said multiple call-chains having the **common call site**.

The remaining allegations made in the Examiner's Answer regarding the foregoing claims have already been addressed in the Appeal Brief.

The obviousness rejection of the above listed claims is therefore clearly erroneous.

B. REPLY TO EXAMINER'S ANSWER REGARDING THE § 103 REJECTION OF CLAIMS 2, 3, 8, 9

Dependent claim 2 further recites that whenever a call site from routine x to routine y is inlined, **new call sites are added from routine x to all routines inlinable within routine y**. .

The Examiner's Answer cited the example shown in Fig. A7-3 of Kramskoy as purportedly depicting the foregoing feature of claim 2. Specifically, the Examiner argued that the "Second Section of Code" shown in Fig. A7-3 of Kramskoy is "routine x" of claim 2, and that the method bar is "routine y" of claim 2. In the example of Fig. A7-3, the call to method bar is inlined into the Second Section of Code. The Examiner argued that "4034 is added from Second Section of Code (x) to all routines/instructions within inlined "bar" (y)." Examiner's Answer at 16. This statement is incorrect. Once the call to the method bar is inlined into the Second Section of Code, the code for bar is actually incorporated in the Second Section of Code, as shown in Fig. A7-3. There is absolutely no reason to add call sites to the method bar that has been incorporated into the Second Section of Code, as alleged by the Examiner.

In view of the foregoing, and in view of the arguments presented in the Appeal Brief, it is respectfully submitted that the foregoing claims are further allowable for the foregoing reason.

Therefore, reversal of the rejection of the above claims is respectfully requested.

C. REPLY TO EXAMINER'S ANSWER REGARDING THE § 103 REJECTION OF CLAIMS 4-6, 10-12

Dependent claim 4 depends from claim 2 (indirectly) and therefore is allowable for at least the same reasons as claim 2.

Claim 4 further recites adding the new call sites to a global work-list so that the new call sites are considered for inlining. The Examiner cited the "dispatch table" mentioned in column 85, lines 58-59, of Kramskoy as being the "global work-list" of claim 4. Regardless of whether or not the "dispatch table" can be properly considered a "global work-list," it is noted that the column 85 passage of Kramskoy clearly does not provide any hint of the added new call sites recited in claim 4, for reasons given above with respect to claim 2.

In view of the foregoing, and in view of the arguments presented in the Appeal Brief, it is respectfully submitted that the foregoing claims are further allowable for the foregoing reason.

Therefore, reversal of the rejection of the above claims is respectfully requested.

D. CONCLUSION

In view of the arguments presented above and in view of the arguments presented in the Appeal Brief, reversal of all final rejections is respectfully requested.

Respectfully submitted,

Date: November 17, 2009

/Dan C. Hu /

Dan C. Hu
Registration No. 40,025
TROP, PRUNER & HU, P.C.
1616 South Voss Road, Suite 750
Houston, TX 77057-2631
Telephone: (713) 468-8880
Facsimile: (713) 468-8883